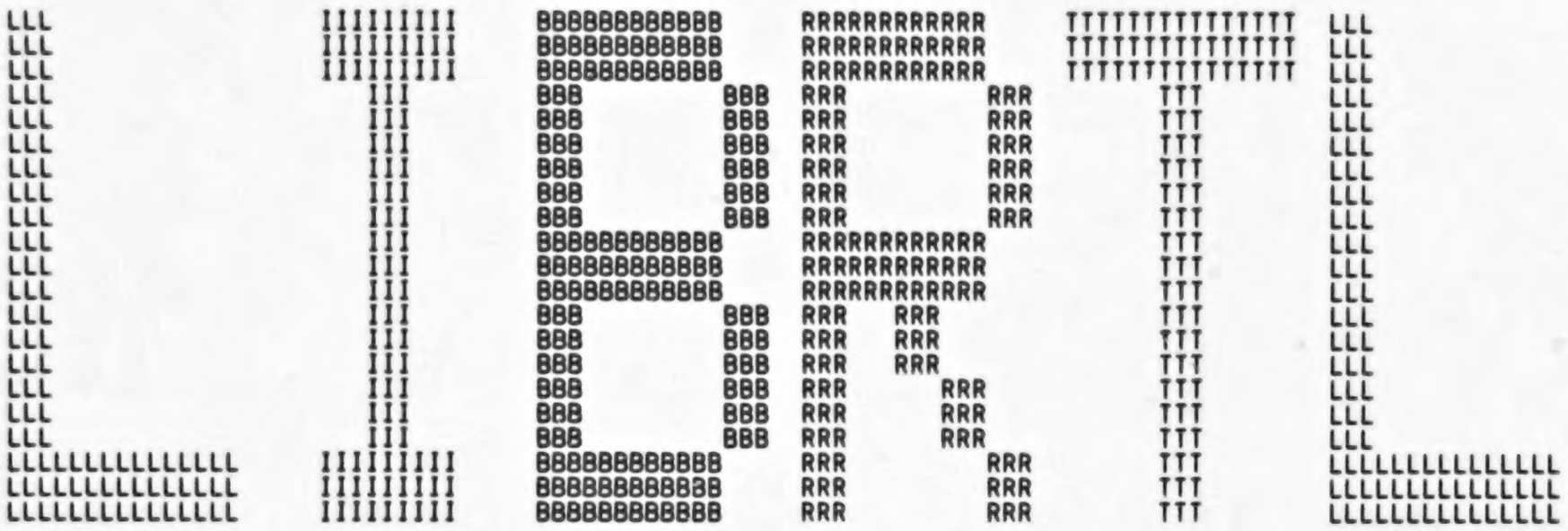
LIBRARY

FILEID**LIBVM

```
LL            IIIIII    BBBBBBBB    VV      VV  MM      MM
LL            IIIIII    BBBBBBBB    VV      VV  MM      MM
LL              II      BB      BB  VV      VV  MMMM  MMMM
LL              II      BB      BB  VV      VV  MMMM  MMMM
LL              II      BB      BB  VV      VV  MM  MM  MM
LL              II      BB      BB  VV      VV  MM  MM  MM
LL              II      BBBBBBBB    VV      VV  MM      MM
LL              II      BBBBBBBB    VV      VV  MM      MM
LL              II      BB      BB  VV      VV  MM      MM
LL              II      BB      BB  VV      VV  MM      MM
LL              II      BB      BB    VV  VV    MM      MM      ....
LL              II      BB      BB    VV  VV    MM      MM      ....
LLLLLLLLLL    IIIIII    BBBBBBBB        VV      MM      MM      ....
LLLLLLLLLL    IIIIII    BBBBBBBB        VV      MM      MM      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE LIB$VM (                                       ! Virtual memory allocation/deallocation
;     2          0002  0                 IDENT = '2-046'                       ! File: LIBVM.B32  Edit: RKR2046
;     3          0003  0                 ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !**************************************************************************
;     7          0007  1 !*                                                                        *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                  *
;    11          0011  1 !*                                                                        *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                           *
;    18          0018  1 !*                                                                        *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21          0021  1 !*  CORPORATION.                                                           *
;    22          0022  1 !*                                                                        *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !**************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY: Resource allocation library
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:  Dynamic virtual memory allocation and deallocation.
;    34          0034  1 !
;    35          0035  1 !       Dynamic virtual memory allocation and deallocation.
;    36          0036  1 !       This facility is the only user mode procedure for allocating
;    37          0037  1 !       and deallocation virtual memory.  By having all procedures use
;    38          0038  1 !       this facility, allocation conflict is eliminated.
;    39          0039  1 !
;    40          0040  1 ! ENVIRONMENT: User access mode; mixture of AST level or not.
;    41          0041  1 !
;    42          0042  1 ! AUTHOR:       Trevor J. Porter, CREATION DATE: 14-Jan-77; Version 01
;    43          0043  1 !
;    44          0044  1 ! MODIFIED BY:
;    45          0045  1 !
;    46          0046  1 !       Thomas N. Hastings, 31-may-77: Version 02
;    47          0047  1 ! 01    - original in linker
;    48          0048  1 ! 02-10 - Add new entry point names LIB$GET_VM, LIB$FREE_VM.  TNH 8-Oct-77
;    49          0049  1 ! 02-15 - Use RTLMSG error codes.  TNH 21-Nov-77
;    50          0050  1 ! 02-16 - Change LIB$_NORMAL to LIB$_NORMAL.  TNH 21-Nov-77
;    51          0051  1 ! 02-17 - Don't clear memory.  TNH 19-Dec-77.
;    52          0052  1 ! 02-18 - Remove LIB$VM_GET, LIB$VM_RET entry points.  TNH 30_Jan-78
;    53          0053  1 ! 02-19 - Change expand size to 128., keep track of largest area
;    54          0054  1 !         allocated so far for validity check in FREE_VM.  JMT 5-Mar-78
;    55          0055  1 ! 02-22 - Change REQUIRE files for VAX system build.  DGP 28-Apr-78
;    56          0056  1 ! 02-23 - Return SS$_NORMAL instead of LIB$_NORMAL.  TNH 15-July-78
;    57          0057  1 ! 02-24 - Use partial allocation from $EXPREG.  TNH 29-July-78
```

```
;    58          0058  1 ! 02-25 - Don't initialize MINADDRESS.  TNH 31-July-78
;    59          0059  1 ! 02-30 - Make AST re-entrant.  TNH 9-Aug-78
;    60          0060  1 ! 2-034 - Update copyright notice and require file names.  JBS 22-NOV-78
;    61          0061  1 ! 2-035 - Run through PRETTY and put in redundent values to
;    62          0062  1 !         keep the new BLISS compiler happy.  JBS 22-NOV-78
;    63          0063  1 ! 2-036 - Put in routine headers for the internal routines ALLOCATE
;    64          0064  1 !         and DEALLOCATE, remove false comments and generally fix
;    65          0065  1 !         up the format to conform to RTL standards.  JBS 02-APR-1979
;    66          0066  1 ! 2-037 - Make the entry points LIB$$GET_VM and LIB$$FREE_VM, since
;    67          0067  1 !         the string package is taking over the job of allocating
;    68          0068  1 !         and deallocating small amounts of storage.  LIB$$GET_VM
;    69          0069  1 !         will still be called for large amounts of storage;
;    70          0070  1 !         LIB$$FREE_VM will free those large amounts.  JBS 02-APR-1979
;    71          0071  1 ! 2-038 - Correct the consistency check in LIB$$FREE_VM: it was off
;    72          0072  1 !         by 1.  JBS 09-APR-1979
;    73          0073  1 ! 2-039 - Add some comments based on the code review.  JBS 12-JUN-1979
;    74          0074  1 ! 2-040 - Undo edit 037.  JBS 27-JUN-1979
;    75          0075  1 ! 2-041 - Remove the redundent values added in edit 035; the new BLISS compiler
;    76          0076  1 !         doesn't need them.  JBS 06-SEP-1979
;    77          0077  1 ! 2-042 - Add statistics cells for LIB$STAT_VM, and clean up compare operators
;    78          0078  1 !         to use address form when appropriate.  JBS 28-OCT-1979
;    79          0079  1 ! 2-043 - When calling $EXPREG, ask for at least enough pages to fulfil the
;    80          0080  1 !         user's request.  Make MAX_ADDRESS 1 greater than the maximum
;    81          0081  1 !         address allocated so that the compare in LIB$FREE_VM is easier.
;    82          0082  1 !         SBL 14-Aug-1981
;    83          0083  1 ! 2-044 - Add logic to try to alleviate, if not cure, problem whereby
;    84          0084  1 !         caller gets into a pattern of allocating space at Non-AST
;    85          0085  1 !         level and freeing the space at AST level.  Eventually all
;    86          0086  1 !         available space migrates to the AST-level queue and not
;    87          0087  1 !         enough remains available at Non-AST level.  Strategy is for
;    88          0088  1 !         ALLOCATE, before resorting to a $EXPREG, to repeatedly try to
;    89          0089  1 !         pull some space from the AST-level queue (if any is there) and
;    90          0090  1 !         ALLOCATE itself is running at Non-AST level.
;    91          0091  1 !         (In response to QAR 893)
;    92          0092  1 !         RKR 12-JAN-1982
;    93          0093  1 !********** Start of post VMS Version 3.0 changes *****
;    94          0094  1 ! 2-045 - Fix for SPR 11-50075.
;    95          0095  1 !         Logic added in previous change has a missing ELSE clause that
;    96          0096  1 !         can cause DEALLOCATE to be called with undefined arguments.
;    97          0097  1 !         Also fix path through this code whereby it is possible for
;    98          0098  1 !         AST's to be disabled and never renabled.  RKR 15-OCT-1982
;    99          0099  1 ! 2-046 - Decrement NEST_LEVEL on error exits. RKR 11-AUG-1983.
;   100          0100  1 !--
;   101          0101  1
;   102          0102  1 !<BLF/PAGE>
```

```
;  104     0103  1 !
;  105     0104  1 ! SWITCHES:
;  106     0105  1 !
;  107     0106  1
;  108     0107  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;  109     0108  1
;  110     0109  1 !
;  111     0110  1 ! LINKAGES:
;  112     0111  1 !
;  113     0112  1 !       NONE
;  114     0113  1 !
;  115     0114  1 ! TABLE OF CONTENTS:
;  116     0115  1 !
;  117     0116  1
;  118     0117  1 FORWARD ROUTINE
;  119     0118  1     LIB$GET_VM,                         ! Allocate virtual memory - interface
;  120     0119  1     ALLOCATE,                           ! Allocate virtual memory - body
;  121     0120  1     LIB$FREE_VM,                        ! Deallocate virtual memory - interface
;  122     0121  1     DEALLOCATE;                         ! Deallocate virtual memory - body
;  123     0122  1
;  124     0123  1 !
;  125     0124  1 ! INCLUDE FILES:
;  126     0125  1 !
;  127     0126  1
;  128     0127  1 REQUIRE 'RTLIN:RTLPSECT';               ! Define DECLARE_PSECTS macro
;  129     0222  1
;  130     0223  1 LIBRARY 'RTLSTARLE';                    ! System symbols
;  131     0224  1
;  132     0225  1 !
;  133     0226  1 ! MACROS:
;  134     0227  1 !
;  135     0228  1 !       NONE
;  136     0229  1 !
;  137     0230  1 ! EQUATED SYSMBOLS:
;  138     0231  1 !
;  139     0232  1
;  140     0233  1 LITERAL
;  141     0234  1     K_VM_MAX_ADR = %X'40000000',        ! Max. possible address (in control region)
;  142     0235  1     K_EXPAND_SIZE = 128,                ! No. of pages to expand program region each call
;  143     0236  1     K_MAX_NEST_LEV = 4;                 ! Max. re-entrant nest level.
;  144     0237  1
;  145     0238  1                                         ! Must be at least 2 for AST and non-AST level.
;  146     0239  1                                         ! Any more allows some multiplexing from AST level.
;  147     0240  1 !
;  148     0241  1 ! PSECT DECLARATIONS:
;  149     0242  1 !
;  150     0243  1 DECLARE_PSECTS (LIB);                   ! declare PSECTs LIB$ facility
;  151     0244  1 !
;  152     0245  1 ! OWN AND GLOBAL STORAGE:
;  153     0246  1 !
;  154     0247  1
;  155     0248  1 OWN
;  156     0249  1     MIN_ADDRESS : INITIAL (0),          ! Lowest address ever allocated for any nest level
;  157     0250  1                                         ! 0 is special case
;  158     0251  1     MAX_ADDRESS : INITIAL (0),          ! Max address allocated so far for any nest level
;  159     0252  1                                         ! Value is actually the next
;  160     0253  1                                         ! byte after the highest allocated.
```

```
;  161     0254  1                                          ! 0 is special case.
;  162     0255  1 !+
;  163     0256  1 ! Free memory list heads
;  164     0257  1 ! one list for each nest level.
;  165     0258  1 ! 1-origin so 0th entry not used.
;  166     0259  1 !-
;  167     0260  1     Q_LIST_HEAD : VECTOR [K_MAX_NEST_LEV*2 + 2] INITIAL ( REP K_MAX_NEST_LEV + 1 OF (0, 0)),
;  168     0261  1 !+
;  169     0262  1 ! Current re-entrant nest level.
;  170     0263  1 ! Counted up each enrty to LIB$GET_VM or LIB$FREE_VM.
;  171     0264  1 ! Counted down on each exit.
;  172     0265  1 ! Starts at 0, so runs from 1...K_MAX_NEST_LEV.
;  173     0266  1 !-
;  174     0267  1     NEST_LEVEL : INITIAL (0);
;  175     0268  1
;  176     0269  1 !+
;  177     0270  1 ! The following statistical cells are reported by LIB$STAT_VM.
;  178     0271  1 !-
;  179     0272  1
;  180     0273  1 GLOBAL
;  181     0274  1     LIB$$GL_GETVM_C : INITIAL (0),              ! Number of successful calls to LIB$GET_VM
;  182     0275  1     LIB$$GL_FREVM_C : INITIAL (0),              ! Number of successful calls to LIB$FREE_VM
;  183     0276  1     LIB$$GL_VMINUSE : INITIAL (0);              ! Bytes still allocated
;  184     0277  1
;  185     0278  1 !
;  186     0279  1 ! EXTERNAL REFERENCES:
;  187     0280  1 !
;  188     0281  1 !+
;  189     0282  1 ! The following are the error codes used in this module:
;  190     0283  1 !-
;  191     0284  1
;  192     0285  1 EXTERNAL LITERAL
;  193     0286  1     LIB$_BADBLOADR : UNSIGNED (%BPVAL),         ! Bad block address
;  194     0287  1     LIB$_BADBLOSIZ : UNSIGNED (%BPVAL),         ! Bad block size
;  195     0288  1     LIB$_FATERRLIB : UNSIGNED (%BPVAL),         ! Fatal error in library
;  196     0289  1     LIB$_INSVIRMEM : UNSIGNED (%BPVAL);         ! Insufficient virtual memory
;  197     0290  1
```

```
;  199     0291  1 GLOBAL ROUTINE LIB$GET_VM (                   ! Allocate dynamic virtual memory
;  200     0292  1         NUM_BYTES,                             ! Adr. of longword size in bytes
;  201     0293  1         BLK_ADR                                ! Adr. of longword to receive assigned adr.
;  202     0294  1     ) =
;  203     0295  1
;  204     0296  1 !++
;  205     0297  1 ! FUNCTIONAL DESCRIPTION:
;  206     0298  1 !
;  207     0299  1 !       Allocate n virtually contiguous bytes at an arbitrary place in
;  208     0300  1 !       the program region and return the virtual address of the first
;  209     0301  1 !       byte. The number of bytes is rounded up so that the smallest
;  210     0302  1 !       number of whole quad words (8 bytes) are allocated starting at a
;  211     0303  1 !       quad word boundary.  Procedures cannot count on successive calls
;  212     0304  1 !       to allocate adjacent blocks of bytes, since an AST, exception or
;  213     0305  1 !       called procedure could also have asked for virtual memory.
;  214     0306  1 !       Usually, the bytes are allocated at the end of the Program
;  215     0307  1 !       region.  However, if there is a sufficiently large hole, it will
;  216     0308  1 !       be used instead. Should there not be enough virtual memory
;  217     0309  1 !       of the required size, the operating system
;  218     0310  1 !       is called to expand the program region by K_EXPAND_SIZE*512 bytes.
;  219     0311  1 !       The new area is linked (by deallocating it) into the free list
;  220     0312  1 !       and the requested memory is allocated from the free list. The
;  221     0313  1 !       free list is therefore initialized on the first allocaton call.
;  222     0314  1 !       AST and non-AST levels are assigned from differrent pools.
;  223     0315  1 !
;  224     0316  1 ! CALLING SEQUENCE:
;  225     0317  1 !
;  226     0318  1 !       STATUS.WLC.V = LIB$GET_VM  (NUM_BYTES.rlu.r, BLK_ADR.wa.r)
;  227     0319  1 !
;  228     0320  1 ! INPUT PARAMETERS:
;  229     0321  1 !
;  230     0322  1 !       NUM_BYTES is the address of an unsigned  longword  integer
;  231     0323  1 !               specifying the number of virtually contiguous bytes to
;  232     0324  1 !               be allocated.  Sufficient pages are allocated  to
;  233     0325  1 !               satisfy the request. However, the program should not
;  234     0326  1 !               reference before the first byte address assigned
;  235     0327  1 !               (base_address) or beyond the last byte assigned
;  236     0328  1 !               (base_adr+num_bytes - 1) since it may be assigned to
;  237     0329  1 !               another procedure.
;  238     0330  1 !
;  239     0331  1 ! OUTPUT PARAMETERS:
;  240     0332  1 !
;  241     0333  1 !    BLK_ADR the address of a longword which is set to the
;  242     0334  1 !               first virtual address of the newly assigned contiguous
;  243     0335  1 !               block of bytes.
;  244     0336  1 !
;  245     0337  1 ! IMPLICIT INPUTS:
;  246     0338  1 !
;  247     0339  1 !       Own storage is used to keep track of unallocated pages in the
;  248     0340  1 !       program region.  The first call after an image is activated
;  249     0341  1 !       causes the OWN storage to be initialized.
;  250     0342  1 !
;  251     0343  1 ! IMPLICIT OUTPUTS:
;  252     0344  1 !
;  253     0345  1 !       NONE.
;  254     0346  1 !
;  255     0347  1 ! COMPLETION STATUS:
```

```
;  256     0348  1 !
;  257     0349  1 !       SS$_NORMAL indicates normal successful completion.
;  258     0350  1 !       LIB$_INSVIRMEM indicates 'INSUFFICIENT VIRTUAL MEMORY' when the
;  259     0351  1 !               program region was attempted to be expanded.
;  260     0352  1 !       LIB$_BADBLOSIZ indicates 'BAD BLOCK SIZE (0)
;  261     0353  1 !               No partial assignment is made.
;  262     0354  1 !
;  263     0355  1 ! SIDE EFFECTS:
;  264     0356  1 !
;  265     0357  1 !       An appropriate number of virtual bytes are removed from the image
;  266     0358  1 !       free memory list.  If needed the program region is expanded by
;  267     0359  1 !       calling the SYS$EXPREG system service.  After this is done ASTs are
;  268     0360  1 !       disabled for a few instructions to update some OWN storage.
;  269     0361  1 !
;  270     0362  1 !--
;  271     0363  1
;  272     0364  2     BEGIN
;  273     0365  2
;  274     0366  2     LOCAL
;  275     0367  2         STATUS,
;  276     0368  2         L_BLK_SIZE;                             ! size of block in bytes modulo quad word
;  277     0369  2
;  278     0370  2     L_BLK_SIZE = (..NUM_BYTES + 7) AND ( NOT 7);        ! Round up to multiple of 8 bytes
;  279     0371  2 !+
;  280     0372  2 ! If the requested block size is zero, give an error indication.
;  281     0373  2 !-
;  282     0374  2
;  283     0375  2     IF (.L_BLK_SIZE EQL 0) THEN RETURN (LIB$_BADBLOSIZ);
;  284     0376  2
;  285     0377  2 !+
;  286     0378  2 ! Arg ok, increment re-entrant nest level index and select corresponding
;  287     0379  2 ! nest level queue header.  Usually this is level 1, since rare to be
;  288     0380  2 ! called at AST level while in LIB$GET_VM or LIB$FREE_VM at non-AST
;  289     0381  2 ! level.
;  290     0382  2 !-
;  291     0383  2     NEST_LEVEL = .NEST_LEVEL + 1;
;  292     0384  2
;  293     0385  2     IF .NEST_LEVEL GTRU K_MAX_NEST_LEV
;  294     0386  2     THEN
;  295     0387  3         BEGIN ! Too deep
;  296     0388  3         NEST_LEVEL = .NEST_LEVEL - 1;
;  297     0389  3         RETURN (LIB$_FATERRLIB);
;  298     0390  2         END;  ! Too deep
;  299     0391  2
;  300     0392  2 !+
;  301     0393  2 ! Allocate space by removing from corresponding queue for this nest level.
;  302     0394  2 !-
;  303     0395  2     STATUS = ALLOCATE (.L_BLK_SIZE, .BLK_ADR, Q_LIST_HEAD [.NEST_LEVEL*2]);
;  304     0396  2 !+
;  305     0397  2 ! Now count re-entrant nest level back down.
;  306     0398  2 ! Usually this just goes from 1 back to 0.
;  307     0399  2 !-
;  308     0400  2     NEST_LEVEL = .NEST_LEVEL - 1;
;  309     0401  2     RETURN (.STATUS);
;  310     0402  1     END;                                        !  end of LIB$GET_VM routine
```

```
                                        .TITLE  LIB$VM
                                        .IDENT  \2-046\

                                        .PSECT  _LIB$DATA,NOEXE,  PIC,2

                        00000000  00000 MIN_ADDRESS:
                                        .LONG   0                                       ;
                        00000000  00004 MAX_ADDRESS:
                                        .LONG   0                                       ;
                00000000 00000000  00008 Q_LIST_HEAD:
                                        .LONG   0, 0                                    ;
                00000000 00000000  00010        .LONG   0, 0                            ;
                00000000 00000000  00018        .LONG   0, 0                            ;
                00000000 00000000  00020        .LONG   0, 0                            ;
                00000000 00000000  00028        .LONG   0, 0                            ;
                        00000000  00030 NEST_LEVEL:
                                        .LONG   0                                       ;
                        00000000  00034 LIB$$GL_GETVM_C::
                                        .LONG   0                                       ;
                        00000000  00038 LIB$$GL_FREVM_C::
                                        .LONG   0                                       ;
                        00000000  0003C LIB$$GL_VMINUSE::
                                        .LONG   0                                       ;

                                        .EXTRN  LIB$_BADBLOADR, LIB$_BADBLOSIZ
                                        .EXTRN  LIB$_FATERRLIB, LIB$_INSVIRMEM

                                        .PSECT  _LIB$CODE,NOWRT,  SHR,  PIC,2

                                0004 00000      .ENTRY  LIB$GET_VM, Save R2              ; 0291
                        52 00000000'  EF  9E 00002      MOVAB   NEST_LEVEL, R2          ;
                50      04    BC      07  C1 00009      ADDL3   #7, @NUM_BYTES, R0      ; 0370
                51            50      07  CB 0000E      BICL3   #7, R0, L_BLK_SIZE      ;
                                      08  12 00012      BNEQ    1$                      ; 0375
                        50 00000000G  8F  D0 00014      MOVL    #LIB$_BADBLOSIZ, R0     ;
                                      04  0001B      RET                              ;
                                      62  D6 0001C 1$:  INCL    NEST_LEVEL              ; 0383
                        04            62  D1 0001E      CMPL    NEST_LEVEL, #4          ; 0385
                                      0A  1B 00021      BLEQU   2$                      ;
                                      62  D7 00023      DECL    NEST_LEVEL              ; 0388
                        50 00000000G  8F  D0 00025      MOVL    #LIB$_FATERRLIB, R0     ; 0389
                                      04  0002C      RET                              ;
                50            62      01  78 0002D 2$:  ASHL    #1, NEST_LEVEL, R0      ; 0395
                                  D8 A240  DF 00031      PUSHAL  Q_LIST_HEAD[R0]         ;
                                  08   AC  DD 00035      PUSHL   BLK_ADR                 ;
                                       51  DD 00038      PUSHL   L_BLK_SIZE              ;
                        0000V     CF        03  FB 0003A      CALLS   #3, ALLOCATE      ;
                                            62  D7 0003F      DECL    NEST_LEVEL        ; 0400
                                            04  00041      RET                         ; 0402

; Routine Size:  66 bytes,    Routine Base:  _LIB$CODE + 0000

;  311          0403  1
```

```
;  313     0404  1 ROUTINE ALLOCATE (                          ! Internal allocation subroutine
;  314     0405  1         SIZE,                               ! Number of bytes to allocate
;  315     0406  1         ADDRESS,                            ! Store base address here
;  316     0407  1         LISTHEAD                            ! Free list for this level
;  317     0408  1     ) =
;  318     0409  1
;  319     0410  1 !++
;  320     0411  1 ! FUNCTIONAL DESCRIPTION:
;  321     0412  1 !
;  322     0413  1 !       Allocate storage from the given list.  If the list does not
;  323     0414  1 !       contain any piece big enough to satisfy the request, expand
;  324     0415  1 !       the program region.
;  325     0416  1 !
;  326     0417  1 ! INPUT PARAMETERS:
;  327     0418  1 !
;  328     0419  1 !       SIZE.rl.v       The number of bytes to allocate.  This is always
;  329     0420  1 !                       a multiple of 8.
;  330     0421  1 !       LISTHEAD.ra.v   The beginning of the list of free blocks at this
;  331     0422  1 !                       reentrancy level.  This list is linked by its
;  332     0423  1 !                       first longword.
;  333     0424  1 !
;  334     0425  1 ! OUTPUT PARAMETERS:
;  335     0426  1 !
;  336     0427  1 !       ADDRESS.wa.r    The address of the block allocated, or 0.
;  337     0428  1 !
;  338     0429  1 ! IMPLICIT INPUTS:
;  339     0430  1 !
;  340     0431  1 !       NONE
;  341     0432  1 !
;  342     0433  1 ! IMPLICIT OUTPUTS:
;  343     0434  1 !
;  344     0435  1 !       NONE.
;  345     0436  1 !
;  346     0437  1 ! COMPLETION STATUS:
;  347     0438  1 !
;  348     0439  1 !       SS$_NORMAL indicates normal successful completion.
;  349     0440  1 !       LIB$_INSVIRMEM indicates 'INSUFFICIENT VIRTUAL MEMORY' when the
;  350     0441  1 !               program region was attempted to be expanded.
;  351     0442  1 !
;  352     0443  1 ! SIDE EFFECTS:
;  353     0444  1 !       An appropriate number of virtual bytes are removed from the image
;  354     0445  1 !       free memory list.  If needed the program region is expanded by
;  355     0446  1 !       calling the SYS$EXPREG system service.
;  356     0447  1 !
;  357     0448  1 !--
;  358     0449  1
;  359     0450  2     BEGIN
;  360     0451  2
;  361     0452  2     LOCAL
;  362     0453  2         GOT_SPACE,                          ! logical to record
;  363     0454  2                                             ! whether we got space
;  364     0455  2                                             ! from other queue
;  365     0456  2         NEWBLOCK : REF VECTOR [2],          ! Current block pointer
;  366     0457  2         NEXTBLOCK : REF VECTOR [2],         ! Next block pointer
;  367     0458  2         LASTBLOCK : REF VECTOR [2],         ! Previous block pointer
;  368     0459  2         MEMLIMITS : VECTOR [2],             ! args to $EXPREG
;  369     0460  2         AST_STATUS;                         ! AST enable state
```

```
;  370     0461  2
;  371     0462  2 !+
;  372     0463  2 ! The following loop is terminated by one of several RETURN statements.
;  373     0464  2 !-
;  374     0465  2
;  375     0466  2     WHILE -1 DO
;  376     0467  3         BEGIN
;  377     0468  3         LASTBLOCK = .LISTHEAD;                  ! Initially at top of free list
;  378     0469  3 !+
;  379     0470  3 ! The following loop scans down the free list looking for a free block
;  380     0471  3 ! which will satisfy the request.  If it finds one it deallocates it
;  381     0472  3 ! and returns.  Otherwise it falls into the next section of code which
;  382     0473  3 ! will attempt to expand the program region.
;  383     0474  3 !-
;  384     0475  3
;  385     0476  3         WHILE (NEWBLOCK = .LASTBLOCK [0]) NEQA 0 DO     ! Follow down free list
;  386     0477  4             BEGIN
;  387     0478  4
;  388     0479  5             IF (.NEWBLOCK [1] EQLU .SIZE)       ! Look for suitable free block
;  389     0480  4             THEN                                ! Exact size match
;  390     0481  5                 BEGIN
;  391     0482  5                 LASTBLOCK [0] = .NEWBLOCK [0];  ! So last points where this one pointed
;  392     0483  5                 .ADDRESS = NEWBLOCK [0];
;  393     0484  5                 LIB$$GL_GETVM_C = .LIB$$GL_GETVM_C + 1;
;  394     0485  5                 LIB$$GL_VMINUSE = .LIB$$GL_VMINUSE + .SIZE;
;  395     0486  5                 RETURN (SS$_NORMAL);            ! and we are done
;  396     0487  4                 END;
;  397     0488  4
;  398     0489  5             IF (.NEWBLOCK [1] GTRU .SIZE)       ! Larger than requested
;  399     0490  4             THEN
;  400     0491  5                 BEGIN
;  401     0492  5 !+
;  402     0493  5 ! We have found a block larger than the size requested.  Divide it in
;  403     0494  5 ! two, with the front used to satisfy the request and the back remaining
;  404     0495  5 ! on the free list.
;  405     0496  5 !-
;  406     0497  5                 NEXTBLOCK = NEWBLOCK [0] + .SIZE;
;  407     0498  5                 NEXTBLOCK [0] = .NEWBLOCK [0];
;  408     0499  5                 NEXTBLOCK [1] = .NEWBLOCK [1] - .SIZE;
;  409     0500  5                 LASTBLOCK [0] = NEXTBLOCK [0];
;  410     0501  5                 .ADDRESS = NEWBLOCK [0];
;  411     0502  5                 LIB$$GL_GETVM_C = .LIB$$GL_GETVM_C + 1;
;  412     0503  5                 LIB$$GL_VMINUSE = .LIB$$GL_VMINUSE + .SIZE;
;  413     0504  5                 RETURN (SS$_NORMAL);            ! and we are done
;  414     0505  4                 END;
;  415     0506  4
;  416     0507  4             LASTBLOCK = NEWBLOCK [0];           ! When not suitable this block becomes previous block
;  417     0508  4             END;                                ! of while loop
;  418     0509  3
;  419     0510  3 !+
;  420     0511  3 ! If we reach this point we know that there is not enough contiguous
;  421     0512  3 ! space in the queue pointed to by the current queue header.  Before
;  422     0513  3 ! resorting to an $EXPREG we check:
;  423     0514  3 !       1. Is there any space in the AST-level queue ?
;  424     0515  3 !       2. Are we ourselves at non-AST level ?
;  425     0516  3 ! If both are true, then we may be able to resolve our problem by
;  426     0517  3 ! moving some space from the AST-level queue to the Non-AST level queue.
```

```
;  427     0518  3 ! If we are at non-AST level (NEST_LEVEL = 1) then we don't have to
;  428     0519  3 ! worry about messing up some interrupted queue manipulation.  However,
;  429     0520  3 ! we must protect ourselves from being interrupted during the critical
;  430     0521  3 ! operation of removing a queue entry from the AST-level queue.
;  431     0522  3 !-
;  432     0523  3
;  433     0524  3     GOT_SPACE = 0 ;                 ! Initialize to got no space
;  434     0525  4     IF ( .Q_LIST_HEAD [4] NEQ 0 )
;  435     0526  3     THEN
;  436     0527  4         BEGIN   ! There was space in AST-level
;  437     0528  4         !+
;  438     0529  4         ! Disable AST's while we figure out if we are at AST level and
;  439     0530  4         ! if so, while we pull off 1st entry of AST-level queue.
;  440     0531  4         !-
;  441     0532  4         AST_STATUS = $SETAST (ENBFLG = 0) ; ! Disable ASTs
;  442     0533  5         IF ( .Q_LIST_HEAD [4] NEQ 0 )           ! Still avail. after
;  443     0534  4         THEN                                    ! disabling AST's ?
;  444     0535  5             BEGIN       ! Safe to proceed
;  445     0536  6             IF ( .NEST_LEVEL EQL 1 )
;  446     0537  5             THEN
;  447     0538  6                 BEGIN   ! We're at non-AST level
;  448     0539  6                 MEMLIMITS [0] = .Q_LIST_HEAD [4] ; ! addr of 1st chunk
;  449     0540  6                 MEMLIMITS [1] = .(.MEMLIMITS [0] + 4) ; ! size of chunk
;  450     0541  6                 Q_LIST_HEAD [4] = ..Q_LIST_HEAD [4] ; ! 1st off head
;  451     0542  6                 GOT_SPACE = 1 ; ! record fact we got space
;  452     0543  5                 END ;   ! We're at non-AST level
;  453     0544  5
;  454     0545  5             !+
;  455     0546  5             ! Renable ASTs whether we succeeded or failed to get space.
;  456     0547  5             !-
;  457     0548  5             IF (.AST_STATUS EQL SS$_WASSET) THEN $SETAST ( ENBFLG = 1) ;
;  458     0549  5
;  459     0550  5             IF .GOT_SPACE       ! If we succeded
;  460     0551  5             THEN
;  461     0552  6                 BEGIN   ! Dump space in out pool of avail. space
;  462     0553  6                 !+
;  463     0554  6                 ! Put this chunk of space on non-AST level queue as if
;  464     0555  6                 ! we had gotten it from $EXPREG.
;  465     0556  6                 !-
;  466     0557  7                 IF ( NOT DEALLOCATE ( .MEMLIMITS [1], ! size of chunk
;  467     0558  7                                       .MEMLIMITS [0], ! address of chunk
;  468     0559  7                                       .LISTHEAD ) )
;  469     0560  6                 THEN
;  470     0561  6                     RETURN (LIB$_FATERRLIB) ; ! Should never happen
;  471     0562  6                 !+
;  472     0563  6                 ! Must back out the modifications made to the statistic
;  473     0564  6                 ! cells by DEALLOCATE.
;  474     0565  6                 !-
;  475     0566  6                 LIB$$GL_VMINUSE = .LIB$$GL_VMINUSE +  .MEMLIMITS [1]  ;
;  476     0567  6                 LIB$$GL_FREVM_C = .LIB$$GL_FREVM_C - 1 ;
;  477     0568  5                 END ;   ! Dump space in our pool of avail. space
;  478     0569  5             END    ! Safe to proceed
;  479     0570  4         ELSE
;  480     0571  5             BEGIN       ! Space disappeared between 1st and 2nd look
;  481     0572  5             ! Reenable ast's if they were enabled.
;  482     0573  5             IF (.AST_STATUS EQL SS$_WASSET) THEN $SETAST ( ENBFLG = 1) ;
;  483     0574  4             END ;       ! Space disappeared between 1st and 2nd look
```

```
;  484     0575  3          END ;   ! There was space in AST-level
;  485     0576  3
;  486     0577  4      IF (NOT .GOT_SPACE)          ! If code above failed to produce more
;  487     0578  3      THEN                         ! space
;  488     0579  3
;  489     0580  4          BEGIN   ! do $EXPREG
;  490     0581  4  !+
;  491     0582  4  ! At this point we have reached the end of the free
;  492     0583  4  ! memory list without finding a block of required size and no more can
;  493     0584  4  ! be liberated from the AST-level queue.
;  494     0585  4  ! Thus, we expand the address space and attempt to
;  495     0586  4  ! allocate from additional virtual memory.
;  496     0587  4  ! If we only get partial allocation, use what we can get.
;  497     0588  4  ! MEMLIMITS[0] is the first virtual address assigned,
;  498     0589  4  ! and MEMLIMITS[1] is the highest virtual address in last page assigned.
;  499     0590  4  ! Both are -1 if nothing was able to be assigned.
;  500     0591  4  !-
;  501   P 0592  4          $EXPREG (PAGCNT = (IF .SIZE LSSU K_EXPAND_SIZE*512 THEN K_EXPAND_SIZE
;  502   P 0593  4                                                          ELSE (.SIZE/512)+1),
;  503     0594  4                   RETADR = MEMLIMITS);
;  504     0595  4
;  505     0596  5          IF (.MEMLIMITS [0] LSS 0)
;  506     0597  4          THEN                                  ! Unsuccessfully expanded program region
;  507     0598  4              RETURN (LIB$_INSVIRMEM);
;  508     0599  4
;  509     0600  4  !+
;  510     0601  4  ! Now disable ASTs and update minimum and maximum addresses ever allocated.
;  511     0602  4  !-
;  512     0603  4          AST_STATUS = $SETAST (ENBFLG = 0);
;  513     0604  4
;  514     0605  4          IF ((.MEMLIMITS [0] LSSA .MIN_ADDRESS) OR (.MIN_ADDRESS EQL 0)) THEN MIN_ADDRESS = .MEMLIMITS [0];
;  515     0606  4
;  516     0607  4          IF ((.MEMLIMITS [1] GTRA .MAX_ADDRESS) OR (.MAX_ADDRESS EQL 0)) THEN MAX_ADDRESS = .MEMLIMITS [1] +
;  517     0608  4
;  518     0609  4          IF (.AST_STATUS EQL SS$_WASSET) THEN $SETAST (ENBFLG = 1);
;  519     0610  4
;  520     0611  4  !+
;  521     0612  4  ! Deallocate the space acquired, thus putting it in the free list.
;  522     0613  4  ! Don't disturb the statistics cells.
;  523     0614  4  !-
;  524     0615  4
;  525     0616  5          IF ( NOT DEALLOCATE ((.MEMLIMITS [1] - .MEMLIMITS [0]) + 1, .MEMLIMITS [0], LASTBLOCK [0]))
;  526     0617  4          THEN
;  527     0618  4              RETURN (LIB$_FATERRLIB);          ! should never happen
;  528     0619  4
;  529     0620  4          LIB$$GL_VMINUSE = .LIB$$GL_VMINUSE + (.MEMLIMITS [1] - .MEMLIMITS [0]) + 1;
;  530     0621  4          LIB$$GL_FREVM_C = .LIB$$GL_FREVM_C - 1;
;  531     0622  3          END;                ! do $EXPREG
;  532     0623  3  !+
;  533     0624  3  ! Now we loop back to search the free list again
;  534     0625  3  !-
;  535     0626  2          END;                                  ! Of WHILE -1 loop
;  536     0627  2
;  537     0628  2      RETURN (LIB$_FATERRLIB);
;  538     0629  1      END;                                      ! of ALLOCATE routine
```

```
                                                       .EXTRN  SYS$SETAST, SYS$EXPREG

                                   07FC 00000 ALLOCATE:
                                                       .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10              ; 0404
                    5A 00000000G   00   9E 00002       MOVAB   SYS$SETAST, R10                               ;
                    59 00000000'   EF   9E 00009       MOVAB   LIB$GL_VMINUSE, R9                            ;
                    5E             08   C2 00010       SUBL2   #8, SP                                        ;
                    55       04    AC   D0 00013       MOVL    SIZE, R5                                      ; 0479
                    56       0C    AC   D0 00017 1$:   MOVL    LISTHEAD, LASTBLOCK                           ; 0468
                    53             66   D0 0001B 2$:   MOVL    (LASTBLOCK), NEWBLOCK                         ; 0476
                                   30   13 0001E       BEQL    6$                                            ;
                    55       04    A3   D1 00020       CMPL    4(NEWBLOCK), R5                               ; 0479
                                   05   12 00024       BNEQ    3$                                            ;
                    66             63   D0 00026       MOVL    (NEWBLOCK), (LASTBLOCK)                       ; 0482
                                   12   11 00029       BRB     4$                                            ; 0483
                                   1E   1B 0002B 3$:   BLEQU   5$                                            ; 0489
           54       53             55   C1 0002D       ADDL3   R5, NEWBLOCK, NEXTBLOCK                       ; 0497
                    64             63   D0 00031       MOVL    (NEWBLOCK), (NEXTBLOCK)                       ; 0498
  04       A4       04    A3       55   C3 00034       SUBL3   R5, 4(NEWBLOCK), 4(NEXTBLOCK)                 ; 0499
                    66             54   D0 0003A       MOVL    NEXTBLOCK, (LASTBLOCK)                        ; 0500
                08  BC             53   D0 0003D 4$:   MOVL    NEWBLOCK, @ADDRESS                            ; 0501
                             F8    A9   D6 00041       INCL    LIB$GL_GETVM_C                                ; 0502
                    69             55   C0 00044       ADDL2   R5, LIB$GL_VMINUSE                            ; 0503
                    50             01   D0 00047       MOVL    #1, R0                                        ; 0504
                                        04 0004A       RET                                                   ;
                    56             53   D0 0004B 5$:   MOVL    NEWBLOCK, LASTBLOCK                           ; 0507
                                   CB   11 0004E       BRB     2$                                            ; 0476
                                   57   D4 00050 6$:   CLRL    GOT_SPACE                                     ; 0524
                             DC    A9   D5 00052       TSTL    Q_LIST_HEAD+16                                ; 0525
                                   5A   13 00055       BEQL    11$                                           ;
                                   7E   D4 00057       CLRL    -(SP)                                         ; 0532
                    6A             01   FB 00059       CALLS   #1, SYS$SETAST                                ;
                    58             50   D0 0005C       MOVL    R0, AST_STATUS                                ;
                    51       DC    A9   D0 0005F       MOVL    Q_LIST_HEAD+16, R1                            ; 0533
                                   42   13 00063       BEQL    10$                                           ;
                    01       F4    A9   D1 00065       CMPL    NEST_LEVEL, #1                                ; 0536
                                   12   12 00069       BNEQ    7$                                            ;
                    6E             51   D0 0006B       MOVL    R1, MEMLIMITS                                 ; 0539
                    50             6E   D0 0006E       MOVL    MEMLIMITS, R0                                 ; 0540
                04  AE       04    A0   D0 00071       MOVL    4(R0), MEMLIMITS+4                            ;
                DC  A9             61   D0 00076       MOVL    (R1), Q_LIST_HEAD+16                          ; 0541
                    57             01   D0 0007A       MOVL    #1, GOT_SPACE                                 ; 0542
                    09             58   D1 0007D 7$:   CMPL    AST_STATUS, #9                                ; 0548
                                   05   12 00080       BNEQ    8$                                            ;
                                   01   DD 00082       PUSHL   #1                                            ;
                    6A             01   FB 00084       CALLS   #1, SYS$SETAST                                ;
                    2D             57   E9 00087 8$:   BLBC    GOT_SPACE, 12$                                ; 0550
                             0C    AC   DD 0008A       PUSHL   LISTHEAD                                      ; 0559
                             04    AE   DD 0008D       PUSHL   MEMLIMITS                                     ; 0558
                             0C    AE   DD 00090       PUSHL   MEMLIMITS+4                                   ; 0557
             0000V  CF             03   FB 00093       CALLS   #3, DEALLOCATE                                ;
                    03             50   E8 00098       BLBS    R0, 9$                                        ;
                                 00A2   31 0009B       BRW     21$                                           ;
                    69       04    AE   C0 0009E 9$:   ADDL2   MEMLIMITS+4, LIB$GL_VMINUSE                   ; 0566
                             FC    A9   D7 000A2       DECL    LIB$GL_FREVM_C                                ; 0567
                                   0A   11 000A5       BRB     11$                                           ; 0533
```

```
                    09              58  D1 000A7 10$:   CMPL    AST_STATUS, #9                          ; 0573
                                    05  12 000AA       BNEQ    11$
                                    01  DD 000AC       PUSHL   #1
                    6A              01  FB 000AE       CALLS   #1, SYS$SETAST
                    03              57  E9 000B1 11$:   BLBC    GOT_SPACE, 12$                          ; 0577
                                FF60    31 000B4       BRW     1$
                                    7E  7C 000B7 12$:   CLRQ    -(SP)                                   ; 0594
                              08    AE  9F 000B9       PUSHAB  MEMLIMITS
                00010000      8F        55  D1 000BC       CMPL    R5, #65536
                                        06  1E 000C3       BGEQU   13$
                    7E          80      8F  9A 000C5       MOVZBL  #128, -(SP)
                                        0C  11 000C9       BRB     14$
              50            55 00000200   8F  C7 000CB 13$:   DIVL3   #512, R5, R0
                                        50  D6 000D3       INCL    R0
                                        50  DD 000D5       PUSHL   R0
                00000000G   00          04  FB 000D7 14$:   CALLS   #4, SYS$EXPREG
                            52          6E  D0 000DE       MOVL    MEMLIMITS, R2                           ; 0596
                                        08  18 000E1       BGEQ    15$
                            50 00000000G    8F  D0 000E3       MOVL    #LIB$_INSVIRMEM, R0                 ; 0598
                                            04  00EA       RET
                                    7E  D4 000EB 15$:   CLRL    -(SP)                                   ; 0603
                    6A              01  FB 000ED       CALLS   #1, SYS$SETAST
                    58              50  D0 000F0       MOVL    R0, AST_STATUS
                C4  A9              52  D1 000F3       CMPL    R2, MIN_ADDRESS                         ; 0605
                                        05  1F 000F7       BLSSU   16$
                              C4    A9  D5 000F9       TSTL    MIN_ADDRESS
                                        04  12 000FC       BNEQ    17$
                C4  A9              52  D0 000FE 16$:   MOVL    R2, MIN_ADDRESS
                C8  A9        04    AE  D1 00102 17$:   CMPL    MEMLIMITS+4, MAX_ADDRESS                ; 0607
                                        05  1A 00107       BGTRU   18$
                              C8    A9  D5 00109       TSTL    MAX_ADDRESS
                                        06  12 0010C       BNEQ    19$
            C8  A9        04    AE      01  C1 0010E 18$:   ADDL3   #1, MEMLIMITS+4, MAX_ADDRESS
                    09              58  D1 00114 19$:   CMPL    AST_STATUS, #9                          ; 0609
                                    05  12 00117       BNEQ    20$
                                    01  DD 00119       PUSHL   #1
                    6A              01  FB 0011B       CALLS   #1, SYS$SETAST
                              0044  8F  BB 0011E 20$:   PUSHR   #^M<R2,R6>                              ; 0616
                52          0C    AE      52  C3 00122       SUBL3   R2, MEMLIMITS+4, R2
                                    01  A2  9F 00127       PUSHAB  1(R2)
                        0000V   CF      03  FB 0012A       CALLS   #3, DEALLOCATE
                                0E      50  E9 0012F       BLBC    R0, 21$
                50              69      52  C1 00132       ADDL3   R2, LIB$$GL_VMINUSE, R0             ; 0620
                                69      01  A0 9E 00136       MOVAB   1(R0), LIB$$GL_VMINUSE
                                    FC  A9  D7 0013A       DECL    LIB$$GL_FREVM_C                     ; 0621
                                        FED7    31 0013D       BRW     1$                              ; 0466
                            50 00000000G    8F  D0 00140 21$:   MOVL    #LIB$_FATERRLIB, R0             ; 0628
                                            04  00147       RET                                         ; 0629

; Routine Size:  328 bytes,    Routine Base:  _LIB$CODE + 0042

;  539          0630  1
```

```
;  542     0631  1 GLOBAL ROUTINE LIB$FREE_VM (                  ! Deallocate virtual memory
;  543     0632  1         NUM_BYTES,                             ! Adr. of longword containing size in bytes
;  544     0633  1         BLK_ADR_ADR                            ! Adr. of longword containing adr. of block
;  545     0634  1     ) =
;  546     0635  1
;  547     0636  1 !++
;  548     0637  1 ! FUNCTIONAL DESCRIPTION:
;  549     0638  1 !
;  550     0639  1 !       Deallocate n virtually contiguous bytes starting at the
;  551     0640  1 !       specified virtual address. The number of bytes actually
;  552     0641  1 !       deallocated is rounded up so that the smallest number of whole
;  553     0642  1 !       quadwords are de-allocated. Numerous error checks are made to
;  554     0643  1 !       make sure that the block being returned is a legitimate free
;  555     0644  1 !       area.
;  556     0645  1 !
;  557     0646  1 ! CALLING SEQUENCE:
;  558     0647  1 !
;  559     0648  1 !       CALL LIB$FREE_VM (NUM_BYTES.rlu.r, BLK_ADR_ADR.ra.r)
;  560     0649  1 !
;  561     0650  1 ! INPUT PARAMETERS:
;  562     0651  1 !
;  563     0652  1 !       NUM_BYTES is the address of an unsigned longword integer
;  564     0653  1 !               specifying the number of virtually contiguous bytes to
;  565     0654  1 !               be deallocated.
;  566     0655  1 !
;  567     0656  1 !       BLK_ADR_ADR is the address of a longword containing the address
;  568     0657  1 !               of the first byte to be deallocated.
;  569     0658  1 !
;  570     0659  1 ! OUTPUT PARAMETERS:
;  571     0660  1 !
;  572     0661  1 !       NONE.
;  573     0662  1 !
;  574     0663  1 ! IMPLICIT INPUTS
;  575     0664  1 !
;  576     0665  1 !       NONE
;  577     0666  1 !
;  578     0667  1 ! IMPLICIT OUTPUTS
;  579     0668  1 !
;  580     0669  1 !       The pages are deallocated by putting them in the list maintained
;  581     0670  1 !       for LIB$GET_VM to search before calling $EXPREG.
;  582     0671  1 !
;  583     0672  1 ! COMPLETION STATUS:                                                    ;
;  584     0673  1 !
;  585     0674  1 !       SS$_NORMAL indicates normal successful completion.
;  586     0675  1 !       LIB$_BADBLOADR indicates BAD BLOCK ADDRESS                      ;
;  587     0676  1 !
;  588     0677  1 ! SIDE EFFECTS:
;  589     0678  1 !
;  590     0679  1 !       Puts the indicated block back on the the image free storage
;  591     0680  1 !       list.
;  592     0681  1 !
;  593     0682  1 !--
;  594     0683  1
;  595     0684  2     BEGIN
;  596     0685  2
;  597     0686  2     LOCAL
;  598     0687  2         STATUS,                                ! Return status
```

```
;  599     0688  2          L_BLK_SIZE;
;  600     0689  2
;  601     0690  2      !+
;  602     0691  2      ! Round up size to be a multiple of quadwords
;  603     0692  2      !-
;  604     0693  2          L_BLK_SIZE = (..NUM_BYTES + 7) AND ( NOT 7);
;  605     0694  2      !+
;  606     0695  2      ! Perform various checks for the validity of the request.
;  607     0696  2      !-
;  608     0697  2
;  609     0698  3          IF (((..BLK_ADR_ADR + .L_BLK_SIZE) GTRA .MAX_ADDRESS) OR (..BLK_ADR_ADR LSSA .MIN_ADDRESS))
;  610     0699  2          THEN
;  611     0700  2              RETURN (LIB$_BADBLOADR);
;  612     0701  2
;  613     0702  2      !+
;  614     0703  2      ! Arg ok, increment re-entrant nest level index and select corresponding
;  615     0704  2      ! nest level queue header.  Usually this is level 1, since need to be
;  616     0705  2      ! called at AST level while in LIB$GET_VM or LIB$FREE_VM at non-AST
;  617     0706  2      ! level.
;  618     0707  2      !-
;  619     0708  2          NEST_LEVEL = .NEST_LEVEL + 1;
;  620     0709  2
;  621     0710  2          IF .NEST_LEVEL GTRU K_MAX_NEST_LEV
;  622     0711  2          THEN
;  623     0712  3              BEGIN    ! Too deep
;  624     0713  3              NEST_LEVEL = .NEST_LEVEL - 1;
;  625     0714  3              RETURN (LIB$_FATERRLIB);
;  626     0715  2              END;     ! Too deep
;  627     0716  2
;  628     0717  2      !+
;  629     0718  2      ! Deallocate space by merging into the corresponding queue for this nest level.
;  630     0719  2      !-
;  631     0720  2          STATUS = DEALLOCATE (.L_BLK_SIZE, ..BLK_ADR_ADR, Q_LIST_HEAD [.NEST_LEVEL*2]);
;  632     0721  2      !+
;  633     0722  2      ! Now count re-entrant nest level back down.
;  634     0723  2      ! Usually this just goes from 1 back to 0.
;  635     0724  2      !-
;  636     0725  2          NEST_LEVEL = .NEST_LEVEL - 1;
;  637     0726  2          RETURN (.STATUS);
;  638     0727  1          END;                                    ! of routine LIB$FREE_VM


                                    0004 00000          .ENTRY  LIB$FREE_VM, Save R2                    ; 0631
                    52 00000000'    EF   9E 00002       MOVAB   NEST_LEVEL, R2
50          04      BC              07   C1 00009       ADDL3   #7, @NUM_BYTES, R0                      ; 0693
51          50                      07   CB 0000E       BICL3   #7, R0, L_BLK_SIZE
50          08      BC              51   C1 00012       ADDL3   L_BLK_SIZE, @BLK_ADR_ADR, R0            ; 0698
            D4      A2              50   D1 00017       CMPL    R0, MAX_ADDRESS
                                    07   1A 0001B       BGTRU   1$
            D0      A2          08  BC   D1 0001D       CMPL    @BLK_ADR_ADR, MIN_ADDRESS
                                    08   1E 00022       BGEQU   2$
                    50 00000000G    8F   D0 00024 1$:   MOVL    #LIB$_BADBLOADR, R0                     ; 0700
                                    04   00028         RET
                                    62   D6 0002C 2$:   INCL    NEST_LEVEL                              ; 0708
```

```
                                04              62  D1 0002E          CMPL    NEST_LEVEL, #4                          ; 0710
                                                0A  1B 00031          BLEQU   3$                                      ;
                                                62  D7 00033          DECL    NEST_LEVEL                              ; 0713
                                50 00000000G    8F  D0 00035          MOVL    #LIB$_FATERRLIB, R0                     ; 0714
                                                    04 0003C          RET                                             ;
                          50                    01  78 0003D 3$:      ASHL    #1, NEST_LEVEL, R0                      ; 0720
                                            D8 A240 DF 00041          PUSHAL  Q_LIST_HEAD[R0]                         ;
                                            08  BC  DD 00045          PUSHL   @BLK_ADR_ADR                            ;
                                                51  DD 00048          PUSHL   L_BLK_SIZE                              ;
                                0000V       CF      03  FB 0004A      CALLS   #3, DEALLOCATE                          ;
                                                    62  D7 0004F      DECL    NEST_LEVEL                              ; 0725
                                                    04 00051          RET                                             ; 0727

; Routine Size:  82 bytes,    Routine Base:  _LIB$CODE + 018A

;  639          0728  1
```

```
;  641     0729  1 ROUTINE DEALLOCATE (                          ! Internal routine to actually deallocate
;  642     0730  1         SIZE,                                  ! The number of bytes to deallocate
;  643     0731  1         ADDRESS,                               ! Base of the area to deallocate
;  644     0732  1         LISTHEAD                               ! List to merge this area into
;  645     0733  1     ) =
;  646     0734  1
;  647     0735  1 !++
;  648     0736  1 ! FUNCTIONAL DESCRIPTION:
;  649     0737  1 !
;  650     0738  1 !       Deallocate storage onto the given list.
;  651     0739  1 !
;  652     0740  1 ! INPUT PARAMETERS:
;  653     0741  1 !
;  654     0742  1 !       SIZE.rl.v       The number of bytes to deallocate.  This is
;  655     0743  1 !                       always a multiple of 8.
;  656     0744  1 !       ADDRESS.ra.r    The address of the block to be deallocated.
;  657     0745  1 !       LISTHEAD.ra.v   The beginning of the list of free blocks at this
;  658     0746  1 !                       reentrancy level.  This list is linked by its
;  659     0747  1 !                       first longword.
;  660     0748  1 !
;  661     0749  1 ! OUTPUT PARAMETERS:
;  662     0750  1 !
;  663     0751  1 !       NONE
;  664     0752  1 !
;  665     0753  1 ! IMPLICIT INPUTS:
;  666     0754  1 !
;  667     0755  1 !       NONE
;  668     0756  1 !
;  669     0757  1 ! IMPLICIT OUTPUTS:
;  670     0758  1 !
;  671     0759  1 !       NONE
;  672     0760  1 !
;  673     0761  1 ! COMPLETION CODES:
;  674     0762  1 !
;  675     0763  1 !       SS$_NORMAL      The deallocation was successful
;  676     0764  1 !       LIB$_BADBLOADR  The block address/length was bad, since it
;  677     0765  1 !                       conflicts with the existing free list.
;  678     0766  1 !
;  679     0767  1 ! SIDE EFFECTS:
;  680     0768  1 !
;  681     0769  1 !       NONE
;  682     0770  1 !
;  683     0771  1 !--
;  684     0772  1
;  685     0773  2     BEGIN
;  686     0774  2
;  687     0775  2     LOCAL
;  688     0776  2         NEWBLOCK : REF VECTOR [2],             ! Current block pointer
;  689     0777  2         NEXTBLOCK : REF VECTOR [2],            ! Next block pointer
;  690     0778  2         LASTBLOCK : REF VECTOR [2];            ! Previous block pointer
;  691     0779  2
;  692     0780  2     LASTBLOCK = .LISTHEAD;                     ! Previous block initially the listhead
;  693     0781  2     NEWBLOCK = .ADDRESS;                       ! Current block is to be inserted
;  694     0782  2 !+
;  695     0783  2 ! Follow down the free list until we reach the end, or the place to
;  696     0784  2 ! insert this block.  The free list is kept sorted so that adjacent
;  697     0785  2 ! free areas can be merged together.
```

```
;  698     0786  2 !-
;  699     0787  2
;  700     0788  2     WHILE ((NEXTBLOCK = .LASTBLOCK [0]) NEQA 0) DO
;  701     0789  3         BEGIN
;  702     0790  3
;  703     0791  4         IF (NEWBLOCK [0] LEQA NEXTBLOCK [0])
;  704     0792  3         THEN
;  705     0793  4             BEGIN
;  706     0794  4 !+
;  707     0795  4 ! This is the position for insertion of the block in the free list.
;  708     0796  4 !-
;  709     0797  4
;  710     0798  5             IF ((NEWBLOCK [0] + .SIZE) EQLA NEXTBLOCK [0])
;  711     0799  4             THEN
;  712     0800  5                 BEGIN                                   ! Here we compact with next block
;  713     0801  5                 NEWBLOCK [0] = .NEXTBLOCK [0];
;  714     0802  5                 NEWBLOCK [1] = .NEXTBLOCK [1] + .SIZE;
;  715     0803  5                 END
;  716     0804  4             ELSE
;  717     0805  5                 BEGIN
;  718     0806  5 !+
;  719     0807  5 ! If this block overlaps the next free block, we have an error.
;  720     0808  5 !-
;  721     0809  5
;  722     0810  5                     IF ((NEWBLOCK [0] + .SIZE) GTRA NEXTBLOCK [0]) THEN RETURN (LIB$_BADBLOADR);
;  723     0811  5
;  724     0812  5                                                         ! BAD BLOCK ADDRESS code
;  725     0813  5                     NEWBLOCK [0] = NEXTBLOCK [0];       ! else set pointer and size since no
;  726     0814  5                     NEWBLOCK [1] = .SIZE;               ! forward compaction needed
;  727     0815  4                     END;
;  728     0816  4
;  729     0817  5             IF (NEWBLOCK [0] EQLA (LASTBLOCK [0] + .LASTBLOCK [1]))
;  730     0818  4             THEN
;  731     0819  5                 BEGIN                                   ! Here we compact with previous
;  732     0820  5                 LASTBLOCK [0] = .NEWBLOCK [0];          ! block
;  733     0821  5                 LASTBLOCK [1] = .NEWBLOCK [1] + .LASTBLOCK [1];
;  734     0822  5                 END
;  735     0823  4             ELSE                                        ! No backward compaction but...
;  736     0824  5                 BEGIN                                   ! must check that block to
;  737     0825  5
;  738     0826  6                 IF (NEWBLOCK [0] LSSA (LASTBLOCK [0] + .LASTBLOCK [1])) ! deallocate is not partially in
;  739     0827  5                 THEN
;  740     0828  5                     RETURN (LIB$_BADBLOADR);            ! previous hole--failure if so
;  741     0829  5
;  742     0830  5                 LASTBLOCK [0] = NEWBLOCK [0];           ! If ok previous points to new one.
;  743     0831  4                 END;                                    ! and we are done compacting
;  744     0832  4
;  745     0833  4             LIB$$GL_FREVM_C = .LIB$$GL_FREVM_C + 1;
;  746     0834  4             LIB$$GL_VMINUSE = .LIB$$GL_VMINUSE - .SIZE;
;  747     0835  4             RETURN (SS$_NORMAL);
;  748     0836  4             END
;  749     0837  3         ELSE
;  750     0838  3             LASTBLOCK = NEXTBLOCK [0];                  ! Not there yet so last block is one just tested
;  751     0839  3
;  752     0840  2         END;                                            ! of WHILE loop
;  753     0841  2
;  754     0842  2 !+
```

```
;  755          0843  2 ! The block to deallocate is beyond the last hole.
;  756          0844  2 ! It must not start within that last hole.
;  757          0845  2 !-
;  758          0846  2
;  759          0847  3     IF (NEWBLOCK [0] LSSA (LASTBLOCK [0] + .LASTBLOCK [1]))
;  760          0848  3     THEN
;  761          0849  3         RETURN (LIB$_BADBLOADR)
;  762          0850  3     ELSE
;  763          0851  3         BEGIN
;  764          0852  3 !+
;  765          0853  3 ! Check to see if the new block goes right after the last old block.
;  766          0854  3 ! If it does we can just extend the last old block.
;  767          0855  3 !-
;  768          0856  3
;  769          0857  4         IF (NEWBLOCK [0] EQLA (LASTBLOCK [0] + .LASTBLOCK [1]))
;  770          0858  3         THEN
;  771          0859  3             LASTBLOCK [1] = .LASTBLOCK [1] + .SIZE
;  772          0860  3         ELSE
;  773          0861  3 !+
;  774          0862  3 ! Otherwise, just put the new block on the end of the free list.
;  775          0863  3 !-
;  776          0864  4             BEGIN
;  777          0865  4             NEWBLOCK [0] = 0;
;  778          0866  4             NEWBLOCK [1] = .SIZE;
;  779          0867  4             LASTBLOCK [0] = NEWBLOCK [0];
;  780          0868  3             END;
;  781          0869  3
;  782          0870  3         LIB$$GL_FREVM_C = .LIB$$GL_FREVM_C + 1;
;  783          0871  3         LIB$$GL_VMINUSE = .LIB$$GL_VMINUSE - .SIZE;
;  784          0872  3         RETURN (SS$_NORMAL);
;  785          0873  2         END;
;  786          0874  2
;  787          0875  1     END;                                    ! of DEALLOCATE routine


                                000C 00000 DEALLOCATE:
                                           .WORD   Save R2,R3                                  ; 0729
                          50       0C AC   D0 00002            MOVL    LISTHEAD, LASTBLOCK    ; 0780
                          51       08 AC   D0 00006            MOVL    ADDRESS, NEWBLOCK      ; 0781
                          53          60   D0 0000A 1$:        MOVL    (LASTBLOCK), NEXTBLOCK ; 0788
                                      42   13 0000D            BEQL    6$
                          53          51   D1 0000F            CMPL    NEWBLOCK, NEXTBLOCK    ; 0791
                                      38   1A 00012            BGTRU   5$
                    52                51       04 AC   C1 00014        ADDL3   SIZE, NEWBLOCK, R2 ; 0798
                          53          52   D1 00019            CMPL    R2, NEXTBLOCK
                                      0C   12 0001C            BNEQ    2$
                          61          63   D0 0001E            MOVL    (NEXTBLOCK), (NEWBLOCK) ; 0801
                04    A1        04  A3       04 AC   C1 00021        ADDL3   SIZE, 4(NEXTBLOCK), 4(NEWBLOCK) ; 0802
                                      0A   11 00028            BRB     3$                     ; 0798
                                      2F   1A 0002A 2$:        BGTRU   7$                     ; 0810
                          61          53   D0 0002C            MOVL    NEXTBLOCK, (NEWBLOCK)  ; 0813
                       04 A1          04   AC C1 0002F        MOVL    SIZE, 4(NEWBLOCK)      ; 0814
                    52                50       04 A0   C1 00034 3$:    ADDL3   4(LASTBLOCK), LASTBLOCK, R2 ; 0817
                          52          51   D1 00039            CMPL    NEWBLOCK, R2           ;
```

```
                                     0A  12 0003C        BNEQ    4$                                          ;
                     60              61  D0 0003E        MOVL    (NEWBLOCK), (LASTBLOCK)                     ; 0820
             04      A0       04     A1  C0 00041        ADDL2   4(NEWBLOCK), 4(LASTBLOCK)                   ; 0821
                                     2E  11 00046        BRB     11$                                         ; 0817
                                     11  1F 00048 4$:    BLSSU   7$                                          ; 0826
                                     27  11 0004A        BRB     10$                                         ; 0830
                     50              53  D0 0004C 5$:    MOVL    NEXTBLOCK, LASTBLOCK                        ; 0838
                                     B9  11 0004F        BRB     1$                                          ; 0788
   52                50       04     A0  C1 00051 6$:    ADDL3   4(LASTBLOCK), LASTBLOCK, R2                 ; 0847
                     52              51  D1 00056        CMPL    NEWBLOCK, R2                                ;
                                     08  1E 00059        BGEQU   8$                                          ;
                     50 00000000G    8F  D0 0005B 7$:    MOVL    #LIB$_BADBLOADR, R0                         ; 0851
                                         04 00062        RET                                                 ;
                                     07  12 00063 8$:    BNEQ    9$                                          ; 0857
             04      A0       04     AC  C0 00065        ADDL2   SIZE, 4(LASTBLOCK)                          ; 0859
                                     0A  11 0006A        BRB     11$                                         ;
                                     61  D4 0006C 9$:    CLRL    (NEWBLOCK)                                  ; 0865
             04      A1       04     AC  D0 0006E        MOVL    SIZE, 4(NEWBLOCK)                           ; 0866
                     60              51  D0 00073 10$:   MOVL    NEWBLOCK, (LASTBLOCK)                       ; 0867
                         00000000'   EF  D6 00076 11$:   INCL    LIB$$GL_FREVM_C                             ; 0870
      00000000'      EF       04     AC  C2 0007C        SUBL2   SIZE, LIB$$GL_VMINUSE                       ; 0871
                     50              01  D0 00084        MOVL    #1, R0                                      ; 0872
                                         04 00087        RET                                                 ; 0875

; Routine Size:  136 bytes,    Routine Base:  _LIB$CODE + 01DC


;  788         0876   1 END                                         ! of LIB$VM module
;  789         0877   1
;  790         0878   0 ELUDOM
```

```
;                                 PSECT SUMMARY
;
;       Name                    Bytes                       Attributes
;
;   _LIB$DATA                      64  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;   _LIB$CODE                     612  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
```

```
;                            Library Statistics
;
;                                          -------- Symbols --------       Pages      Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;   _$255$DUA28:[SYSLIB]STARLET.L32;1       9776         6         0         581       00:00.8
```

```
;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:LIBVM/OBJ=OBJ$:LIBVM MSRC$:LIBVM/UPDATE=(ENH$:LIBVM)

; Size:         612 code + 64 data bytes
; Run Time:       00:10.9
; Elapsed Time:   00:46.0
; Lines/CPU Min:   4850
; Lexemes/CPU-Min: 27939
; Memory Used:  132 pages
; Compilation Complete
```

OTSCCB
LIS

OTSCCBDAT
LIS

OTSCVTDP
LIS

OTSCNVOUT
LIS

OTSCVTFP
LIS

OTSCVTLT
LIS

LIBVECTR2
LIS

LIBWAIT
LIS

OTSCLOSEF
LIS

OTSCVTHP
LIS

LIBVECTOR
LIS

OTSCVTDT
LIS

LIBVM
LIS

OTSCVTGP
LIS